Riku Huuhka

VIRRANKULUTUSMITTAUKSEN AUTOMATISOINTI RANNELAITETESTAUKSESSA

VIRRANKULUTUSMITTAUKSEN AUTOMATISOINTI RANNELAITETESTAUKSESSA

Riku Huuhka
Opinnäytetyö
Kevät 2018
Tietotekniikan tutkinto -ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, älykkäät järjestelmät

Tekijä(t): Riku Huuhka
Opinnäytetyön nimi: Virrankulutusmittauksen automatisointi rannelaitetestauksessa
Työn ohjaaja: Kari Jyrkkä
Työn valmistumislukukausi ja –vuosi: kevät 2018 Sivumäärä: 33

Opinnäytetyön tarkoituksena oli automatisoida nykyistä manuaalisesti tapahtuvaa rannelaitteiden virrankulutustestausta. Työ sisälsi nykytilan kartoituksen, mittalaitteiston valinnan sekä järjestelmän asennuksen. Työn tuloksena syntyi mallitoteutus automatisoidusta rannelaitteen virrankulutustestauksesta, jossa hyödynnettiin yrityksen testiautomaatioympäristöä sekä nykyaikaista mittauslaitteistoa. Työn tilaaja oli Polar Electro Oy ja se toteutettiin yrityksen tiloissa.

Työ alkoi nykytilan kartoituksella, tutustumalla senhetkiseen testaustapaan. Kartoituksen myötä voitiin todeta henkilökuntaa sitovimmat kohdat testausprosessissa ja lähteä suunnittelemaan kehittyneempää testausprosessia, jossa automatisoitaisiin virrankulutustestausta.

Testauksen automatisoinnissa päätettiin hyödyntää yrityksen testiautomaatioympäristöä, jota oli hyödynnetty myös jatkuvan kehityksen tuotetestauksiin. Ympäristön valinta oli perusteltua myös sen ylläpidettävyyden, laajennettavuuden sekä jatkokehitysominaisuuksien vuoksi. Mittauslaitteiston valinnassa huomioitiin erityisesti sen monikäyttöisyys myös tulevaisuudessa. Valintaan vaikuttivat mittauslaitteiston laajennettavuus, mittausalue sekä mittaustarkkuus.

Testitapauksessa, joka on ohjelmoitu Java-kielellä, rannelaitetta ohjataan automatisoidusti suunniteltuun tilaan ja testin aikana suoritetaan myös virrankulutuksen mittaus. Testin aikana rannelaitteelle suoritetaan käyttöönotto, tarkistetaan ja asennetaan viimeisin ohjelmistoversio sekä saatetaan laite haluttuun tilaan mittausta varten. Tässä työssä virrankulutusmittaus suoritetaan rannelaitteen kellotilassa ja virrankulutusta mitataan 33 minuutin ajan, jonka jälkeen tulokset tallentuvat TXT- ja CSV-muodossa. Tuloksena saadaan mittauksen keskiarvoa esittävä tulos.

Jatkokehitysvaiheessa järjestelmän dynaamisuutta voidaan parantaa entisestään hyödyntämällä Jenkins-palvelimia, jotka mahdollistavat ajasta ja paikasta riippumattoman testien suorittamisen. Lisäksi tulokset on mahdollista tuoda yrityksen raportointijärjestelmään automatisoidusti ja laitteisto mahdollistaa usean rannelaitteen samanaikaisen testauksen.

Asiasanat: testiautomaatio, Python, mallitoteutus, virrankulutustestaus

ABSTRACT

Oulu University of Applied Sciences
Degree programme in Information Technology, Intelligent Systems

---

Author(s): Riku Huuhka
Title of thesis: Current consumption automation in wrist unit testing
Supervisor(s): Kari Jyrkkä
Term and year when the thesis was submitted: spring / 2018 Number of pages:33

This Bachelor´s Thesis was about automating current consumption testing in Polar Electro. Objectives of this thesis included present state survey, measuring device selection and system installation. As a result Proof of Concept about automated current consumption system was made. Work was executed at subscriber's premises.

Work started with present state survey about how current consumption tests were executed currently and what would be main improvement points in the process. When all improvement points were approved then followed design phase where the framework of test environment was chosen. Most convenient and economical way to automate current consumption tests was to integrate it to existing test automation environment. Measuring device was selected by the needs for the future where measuring range and accuracy are needed features.

Test case for automated current consumption test was implemented with Java. Test case controls wrist unit to certain state and initiates current consumption measurement. Before current consumption phase starts, test case controls wrist unit to initiate first time use and firmware update. In this work measuring phase lasts 33 minutes and measurement results are reported in TXT and CSV format.

Further development phase includes Jenkins server deployment which enables more flexible measurements in automated environment. Jenkins server also enables automated test result presentation at company´s report program. Premeditated device selection also enables testing of multiple wrist units simultaneously.

Keywords: Test automation, Python, Current measurement, Proof of Concept

SISÄLLYS

1 JOHDANTO

Opinnäytetyön tavoitteena on kehittää rannelaitteiden virrankulutustestausta automatisoimalla nykyisin manuaalisesti tapahtuva testaus sekä testitulosten raportointi niiltä osin, kuin se tällä hetkellä on mahdollista. Työn tarkoituksena on tuoda esille nykyaikaisella mittauslaitteistolla, uudessa testiautomaatioympäristössä suoritettavan virrankulutustestauksen tehokkuus sekä näiden tuomat mahdollisuudet tulevaisuudessa.

Valmis mallitoteutus sisältää testiautomaatioympäristön asennuksen sekä siinä suoritettavan testitapauksen automatisoidulle virrankulutusmittaukselle, valitulla mittalaitteistolla. Mallitoteutuksessa järjestelmän toimivuus todennetaan suorittamalla mittaus rannelaitteen yhdessä tilassa, kellotilassa.

Nykyinen tapa suorittaa testit manuaalisesti on hidas sekä toistettavuudeltaan myös heikko ja vaikuttavat näistä syistä vaihtelevasti myös tuloksiin. Testitulosten kirjaaminen ja tallennus manuaalisesti on myös hyvin aikaa vievää sekä virheille altis tapa ja yhtenä kehityskohteena on automatisoida testitulosten tallennus sekä esitystapa. Mittauksia on tähän asti pystytty suorittamaan ainoastaan yhdelle laitteelle kerrallaan ja työssä selvitetään myös, voidaanko mallitoteutuksessa ohjata useampaa laitetta samanaikaisesti. Valmis työ tullaan ottamaan käyttöön välittömästi nykyisiin tuotekehitysprojekteihin ja testien automatisointia jatketaan.

Virrankulutusta testataan rannelaitteen kaikissa mahdollisissa tiloissa, ja tämän vuoksi jo yhdelle laitteelle tehtävä testimatriisi on hyvin laaja. Rajatakseni testauskohteiden määrän työhöni sopivaksi mallitoteutus kehitetään mittaamalla kellotilan virrankulutusta. Työssä hyödynnetään olemassa olevia rannelaitemalleja, mikä helpottaa tulosten analysointia uudella mittalaitteella ja tuloksia on mahdollista verrata aiemmin saatuihin tuloksiin. Virrankulutuksen testauksessa on tarkoitus hyödyntää mahdollisimman paljon valmiita testiautomaation kirjastoja. Työn tilaaja on Polar Electro Oy ja työ tehdään yhtiön laboratoriotiloissa.

Polar Electro Oy on perustettu vuonna 1977 ja on kehittänyt maailman ensimmäisen langattoman sykemittarin vuonna 1982. Yhtiön pääkonttori sijaitsee Kempeleessä ja sillä on myös toimipisteet Tampereella ja Jyväskylässä. Yhtiön liikevaihto oli (vuosi 2016) 219,4 miljoonaa euroa (1).

Yhtiö on muuntautunut vahvasta laitevalmistajan statuksestaan kokonaisvaltaisen palvelun sekä ohjelmistojen ja laitteiden tuottajaksi. Asiakaskunta on myös laajentunut kilpaurheilijasta enemmän omasta aktiivisuudestaan kiinnostuneisiin arki- sekä aktiiviliikkujiin (2).

2 TYÖN LÄHTÖKOHDAT JA TAVOITTEET

Nykyisissä rannelaitteissa on paljon ominaisuuksia ja ne sisältävät paljon erilaista tekniikkaa. Varsinkin sykemittauksessa optinen sykkeenmittaustekniikka on jo todella yleinen ominaisuus eri laitevalmistajilla. Nykyisiin älykelloihin ja erilaisiin sykemittareihin onkin ahdettu iso määrä erilaisia kiihtyvyysantureita, paineantureita, paikannuslaitteita sekä erilaisia lyhyen kantaman radiolaitteita. Edellä mainituista ominaisuuksista rakentuvat laitteen toiminnallisuudet, joita uudemmissa laitteissa voi olla useita kymmeniä. Kaikkien toiminnallisuuksien pitää luonnollisesti toimia kaikissa mahdollisissa tiloissa ja tilanteissa. Laitteen toiminta varmistetaan suunnittelemalla testaukset huolellisesti niin, että testaussuunnitelmasta muodostuu eräänlainen matriisi laitteen kaikista mahdollisista tilakombinaatioista. Mikäli laitevalmistajalla on lisäksi käytössään laitteeseen liitettävää palvelutoimintaa, on rannelaitteen lisäksi myös testattava rannelaitteen yhteys päätelaitteelle.

Tuotekehityksessä jatkuva virrankulutuksen testaus ja seuranta ovat erittäin tärkeässä roolissa onnistuneen projektin läpiviemisessä. Virrankulutuksen merkitys nousee erityisesti akku- sekä paristokäyttöisissä laitteissa ja on yksi tärkeimmistä ominaisuuksista, joilla pyritään myös erottumaan kilpailijoista. Säännöllisellä virrankulutuksen mittaamisella voidaan arvioida sen hetkistä koodin kypsyyttä ja nopeuttaa myös ongelmien ratkaisua.

Virrankulutusta mitataan laboratorio-olosuhteissa tietyn protokollan mukaisesti ja kirjataan saadut tulokset. Käytännössä mittaukset tehdään aina laitteen uudelle ohjelmistoversioehdokkaalle tai jos halutaan tutkia tarkemmin jotakin tiettyä tilaa rannelaitteesta. Polarin nykyisissäkin rannelaitteissa on paljon eri tiloja ja kaikkien tilojen virran kulutus on testattava.

2.1 Systeemivaatimukset ja tavoitteet

Virrankulutuksen testaaminen tapahtuu tällä hetkellä ainoastaan osittain automaattisesti ja sitoo yhden työntekijän työpanoksen lähes kokonaan testien suorittamiseen. Itse mittaus tapahtuu ilman valvontaa tietokoneohjatusti, mutta mittausta ei esimerkiksi pystytä sammuttamaan automaattisesti tietyn ajan kuluttua. Lisäksi rannelaitteelta mitattava tila on vaihdettava manuaalisesti jokaiseen testiin erikseen ja aloitettava virrankulutusmittaus aina uudestaan, kunnes jokainen tila on testattu.

Toteutuksessa tulee myös huomioida, että työlle annetut vaatimukset täyttyvät. Sen lisäksi että mittalaitteen tulisi soveltua virranmittauksen automatisointiin, sen tulisi olla mahdollisimman monikäyttöinen mittatarkkuuden osalta, jotta samaa järjestelmää voitaisiin hyödyntää tulevaisuudessa mahdollisimman laajasti. Mittalaitteen mittausalueen pitäisi olla vähintään 10 µA – 100 mA ja tarpeen mukaan myös helposti laajennettavissa. Mittausalueen lisäksi laitteessa tulee olla vähintään yksi USB-kanava, jonka avulla rannelaitetta on mahdollista ohjata testin aikana haluttuun tilaan. Mittalaitteen monikäyttöisyyttä tulevaisuudessa lisää myös mitattavien kanavien määrä sekä yhteensopivuus eri käyttöjärjestelmissä.

2.2 Mittalaitteiden vertailu

Nykytilan kartoituksen jälkeen oli tarpeellista perehtyä laitteistovaatimuksiin sekä tutkia eri vaihtoehtoja markkinoilla olevista vaihtoehdoista sekä myös tutkia, voitaisiinko jo olemassa olevia laitteita mahdollisesti hyödyntää työssäni. Yhtenä vaatimuksena on mittalaitteen liitettävyys ja sen ohjattavuus testiautomaatioympäristössä sekä myös laajennettavuus. Mittalaitteen pitää luonnollisesti toimia riittävällä tarkkuustasolla sekä ohjata rannelaitteelle tulevaa USB-yhteyttä.

2.3 National Instruments CompactDAQ (cDAQ-9132) ja YepKit YKUSH3

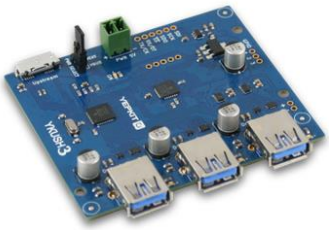
National Instrumentsin mittalaite (kuva 1) mahdollistaa hyvinkin tarkkojen heikkovirtamittausten analysoinnin. Laitteella on mahdollista suorittaa mittauksia tarkkuusalueilla 10 µA – 100 mA, mutta vaatii siihen vielä 24 bitin AD-muuntimen sekä 1 ohmin vastuksen (3). Tässä työssä käytetään kuitenkin 0,9 ohmin vastusta, jolloin pääsemme kuitenkin jo 20 µA:n mittausalueelle. Mittausalueen maksimitasoa on lisäksi mahdollista laajentaa lisäämällä mittalaitteeseen erillinen input moduuli (kuva 2). Tämä mahdollistaa mittaukset 500 mA:iin asti (6).

Laitteen käytössä hyödynnetään G-kielellä ohjelmoitavaa Labview-ohjelmistoa ja sen vuoksi suurin ongelma laitteiston käyttöönotossa on sen liittäminen Java-pohjaiseen testiautomaatioympäristöön. Laitteeseen voidaan kytkeä ainoastaan kaksi USB-laitetta samanaikaisesti ja sen vuoksi laitteeseen pitää kytkeä vielä erillinen USB-jakaja, jotta vaadittava neljän laitteen samanaikainen ohjaaminen onnistuisi. Usean laitteen kytkennässä hyödynnetään YepKit YKUSH 3 -mallista USB-jakajaa (kuva 3). USB-jakaja mahdollistaa USB-yhteyden ohjauksen eli yhteyden katkaisun

testauksen aikana. Laitteen alkaen hinta valmistajan sivujen mukaan on 2630 euroa sekä yhden Yepkitin hinta 79,99 euroa (5).

KUVA 1. National Instruments CompactDAQ (cDAQ-9132) (3)

KUVA 2. National Instruments C Series Voltage Input Module(NI-9238) (4)



KUVA 3. YepKit YKUSH 3 (5)

2.4 Monsoon High Voltage Power Monitor

Monsoon High Voltage Power Monitor -mittalaitetta (kuva 4) on mahdollista ohjata myös Python-skripteillä mutta siihen voidaan liittää ainoastaan yksi laite kerrallaan ja se vaatisi myös lisäksi Yepkitin. Laitteesta löytyy pääkanavan lisäksi USB-kanava sekä AUX-kanava. Laitteen mittaus-tarkkuus olisi kuitenkin 50 µA ja olisi tältä osin soveltuva työhöni. Yhden laitteen hinta on 829 dollaria.

KUVA 4. Monsoon High Voltage Power Monitor (6)

2.5 National Instruments Multifunction I/O Device

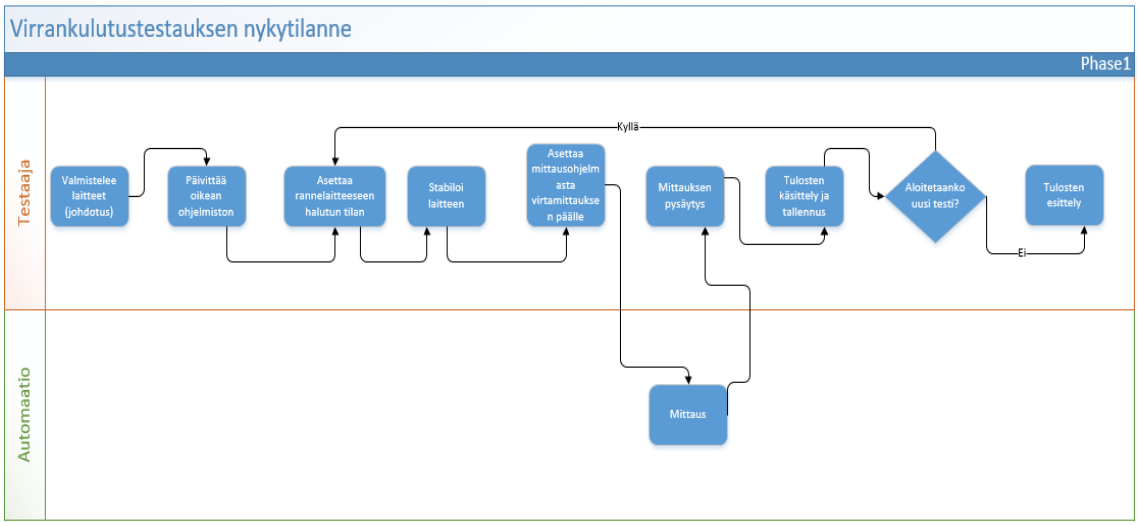
National Instruments Multifunction I/O Device -mittalaite (kuva 5) mahdollistaa monikanavaisen mittaustavan ja sen hyödyt tulevat esille erityisesti sellaisissa tilanteissa, missä pääkanavasta tapahtuva virranmittaus ei ole riittävä. Kyseisellä monikanavaohjaimella on mahdollista tutkia tarkemmin yksittäisen laitekomponentin kuten Bluetoothin, GPS:n tai näytön virrankulutusta. Laitteella on mahdollista mitata neljää kanavaa samanaikaisesti. Saatua dataa voidaan vertailla keskenään ja mahdollinen virtapiikin aiheuttaja on mahdollista saada selvitettyä tällä tavalla tarkasti. Laitteen hinta valmistajan sivuilla on 750 euroa.

KUVA 5. National Instruments Multifunction I/O Device USB-6210 (7)

Koska mitattavia tuotteita on paljon ja virrankulutusta olisi pystyttävä mittamaan mahdollisimman laajasti, paras vaihtoehto mittauksille olisi National Instruments CompactDAQ (cDAQ-9132). Laitteen tarkkuusalue sekä laajennettavuus antavat vaihteleviin mittauksiin merkittävästi dynamiikkaa, ja se on sen vuoksi luonnollinen valinta työssäni käytettäväksi mittalaitteeksi. Kyseisen mittalaitteen lisäksi työssä käytetään vähintään yhtä Yepkitiä.

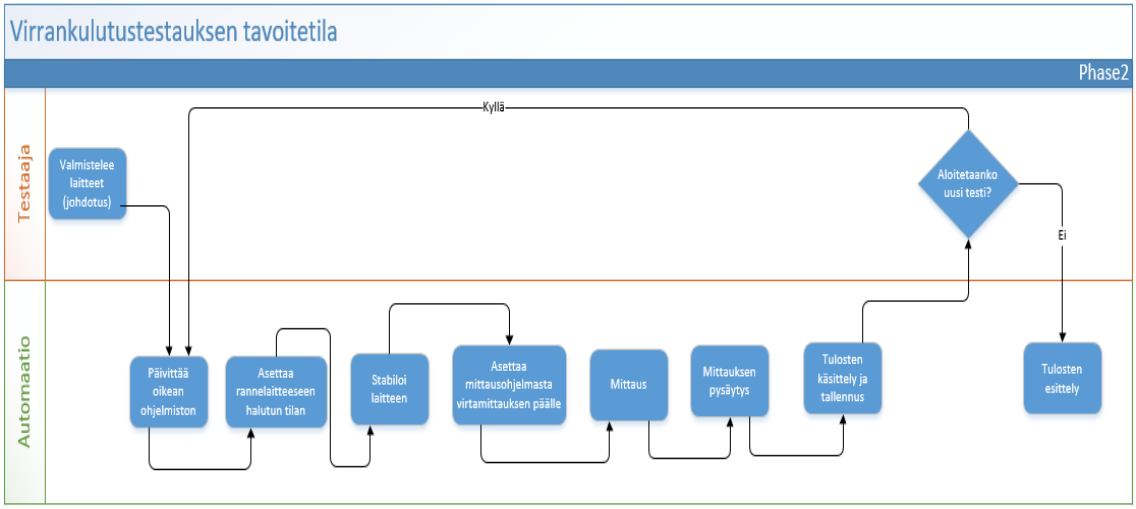
2.6 Virrankulutustestauksen nykytila ja tulevaisuus

Nykyinen virrankulutustestaus sisältää useita manuaalisesti suoritettavia työvaiheita. Ennen testauksen aloittamista mitattava rannelaite valmistellaan johdottamalla se niin, että mittaukset akulta voidaan suorittaa. Tämän jälkeen mitattavasta laitteesta käynnistetään mitattava tila (kellotila, treenitila tms.) päälle ja annetaan sen stabiloitua hetki (1 min) ennen mittauksen käynnistystä. Mittausohjelmasta valitaan mitattavaksi arvoksi avg, koska mittauksessa halutaan ainoastaan virrankulutuksen keskiarvoa esittävä tulos ja mittaus voidaan tämän jälkeen käynnistää. Yhden mittauksen kesto vaihtelee 30 minuutin ja 60 minuutin välillä, jonka jälkeen mittaus voidaan lopettaa ja käsitellä sekä tallentaa saadut tulokset (kuva 6). Tässä oli kuitenkin kuvattuna ainoastaan yhden tuotteelle tehtävän testin prosessi. Testimatriisiin sisältyy tuotteesta riippuen 20–30 yksittäistä mittausta, jotka vaativat aina testaajan työpanoksen edellä kuvatuissa työvaiheissa.



KUVA 6. Virrankulutustestauksen nykytilanne

Virrankulutusta mitataan suhteellisen laajalla testimatriisilla ja jokaisesta mittauksesta tallennetaan saadut tulokset ennen seuraavan testin aloittamista. Manuaalinen tulosten käsittely ja tallentaminen hidastaa koko testausprosessia erittäin paljon. Automatisoidussa ympäristössä testaajasta tai testiympäristöstä johtuvat vaihtelut saadaan minimoitua ja parannettua testien luotettavuutta merkittävästi. (Kuva 7.)



KUVA 7. Virrankulutustestauksen tavoitetila automatisoidussa ympäristössä

Tällä hetkellä rannelaitteiden virrankulutusta mitataan Fluke-yleismittareilla (kuva 8), jotka ovat yhteydessä tietokoneeseen ja siinä suoritettavaan mittausohjelmaan. Mittauksissa hyödynnetään Fluken omaa tietokoneohjelmistoa testien suorittamiseen sekä tulosten tallennukseen.

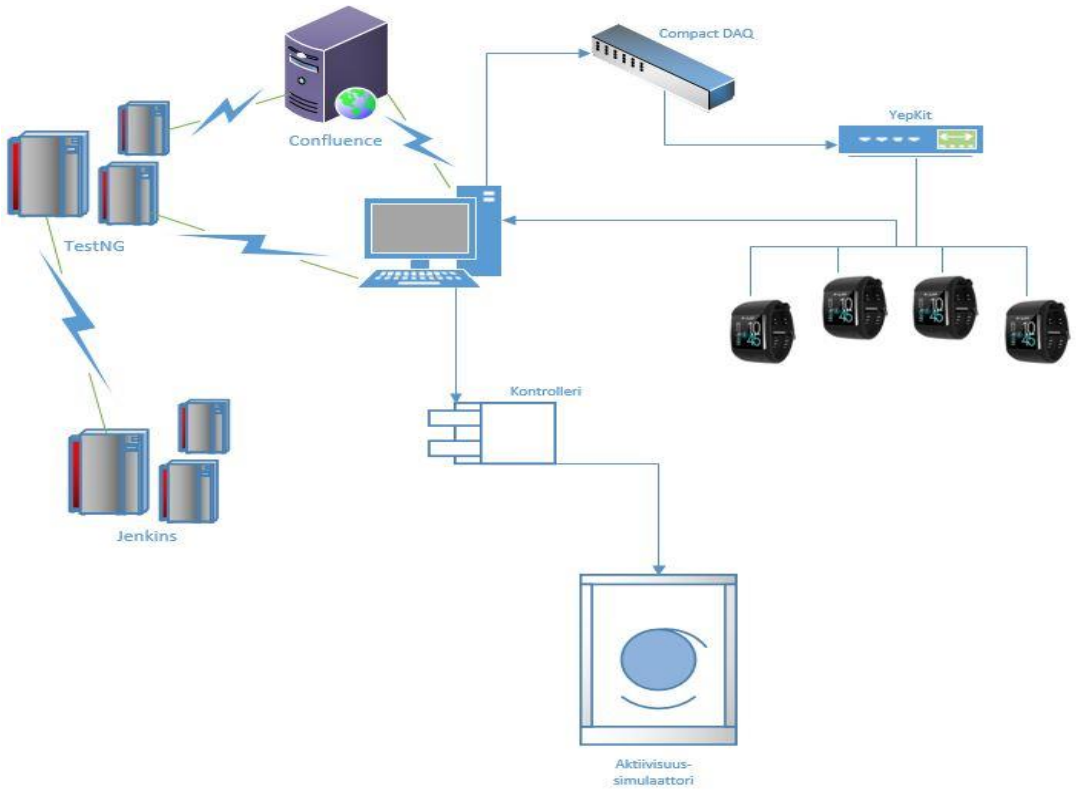
KUVA 8. Nykyinen virranmittauslaitteisto

3 UUSI AUTOMATISOITU VIRRANMITTAUSJÄRJESTELMÄ

Mallitoteutuksessa (kuva 9) hyödynnetään testiautomaatioympäristöä, joka koostuu isosta määrästä rajapintoja muihin sovelluksiin ja ohjelmiin sekä Jenkins-palvelimista ja TestNG-ympäristöstä. Työn kannalta keskeisimmät osat ovat mitattava rannelaite sekä tietokone, johon asennetun kehitysympäristön avulla testitapausta on mahdollista suorittaa. Järjestelmään liitetyt Compact DAQ-mittalaite sekä Yepkit on kytketty tietokoneeseen USB-yhteyden välityksellä. Järjestelmän sekä TestNG-testiautomaatioympäristön luodun rajapinnan avulla testitapauksessa luodaan mitattavalle rannelaitteelle uusi käyttäjä Polar Flow Sync -palvelussa sekä testataan harjoitustietojen synkronointi palveluun.

Kaaviossa kuvattua Jenkins-palvelinta hyödynnetään jatkokehitysvaiheessa ja sillä on kaksi tärkeää tehtävää. Jenkins-palvelimelta käyttäjän on mahdollista käynnistää haluamansa testitapaus sekä määritellä, mihin kellon aikaan testitapaus suoritetaan ja kuinka useasti testitapausta suoritetaan. Tämän lisäksi mittauksesta saatu tulos tallentuu Jenkins-palvelimelle, mistä se on siirrettävissä automaattisesti raportointijärjestelmään.

Arkkitehtuurikaaviossa kuvattuja Confluence raportointijärjestelmää sekä aktiivisuussimulaattoria hyödynnetään vasta järjestelmän jatkokehitysvaiheessa, mutta niiden käyttöönotto oli järkevää huomioida ympäristön asennusvaiheessa.



KUVA 9. Systeemiarkkitehtuurikaavio

TestNG (Next Generation) on Java-pohjainen testausympäristö, joka perustuu paljolti JUnit- ja NUnit-testausympäristöihin. TestNG sisältää paljon parannuksia JUnit-ympäristössä ilmenneisiin rajoituksiin ja ongelmiin ja on tästä syystä paljon tehokkaampi sekä myös käyttäjäystävällisempi testausympäristö. TestNG:llä voidaan suorittaa testausta yksinkertaisista yksikkötestauksista aina monimutkaisempiin integraatiotestauksiin ja sen avulla voidaan myös yksinkertaistaa testausvaatimuksia.

TestNG:n etuina pidetään mm. yksinkertaistettua luokittelua testitapausten luomisessa, mikä puolestaan helpottaa niiden ymmärtämistä. Testiympäristössä on mahdollista suorittaa useita testitapauksia yhdellä kertaa sekä niiden suorittamista on myös mahdollista priorisoida tai sivuuttaa. Testitapauksia on lisäksi mahdollista ryhmitellä ja tulokset on mahdollista generoida HTML-muotoon. (8.)

TestNG voidaan ottaa käyttöön esimerkiksi Eclipse-kehitysympäristössä yksinkertaisella TestNG-liitännäisen asennuksella. Asennuksen onnistuttua Eclipsen valikkoon ilmestyy TestNG-valikot. Testitapausten ohjelmointi tapahtuu normaalissa Java-kehitysympäristössä, kuten Eclipse tai IntelliJ IDEA.

Jenkins on avoimeen lähdekoodiin perustuva automaatiopalvelin, joka mahdollistaa nopeamman ohjelmistokehityksen tarjoamalla yksinkertaisen mallin luoda integrointiympäristöjä lähes kaikille kielille sekä versionhallintatyökaluille. Jenkinsin avulla voidaan myös automatisoida rutiininomaisia kehitystehtäviä. Ohjelmistokehityksessä sitä käytetään lähdekoodin testaukseen sekä uuden ohjelmistoversion kääntämiseen lähdekoodiin tehtyjen muutosten jälkeen. (9.)

Testiautomaatioympäristössä Jenkins on koko toiminnan sydän ja aivan keskiössä koko testauksen ohjaukseen. Jenkinsiä voidaan pitää moottorina, jonka avulla testausta voidaan koordinoida ennalta hyvinkin tarkasti sinne määriteltyjen parametrien avulla. Jenkinsillä voidaan ohjata testauksessa olevia niin sanottuja laitetelineitä niin, että jokaista laitetelineen laitetta voidaan ohjata erikseen aiemmin määriteltyjen parametrien avulla. Parametrit sisältävät tietoja esimerkiksi testattavan ohjelmistoversion sijainnista sekä varsinaisen TestXML-parametrin, joka sisältää tiedot ajettavista testeistä. Jenkinsin avulla testejä voidaan myös suorittaa maantieteellisestä sijainnista riippumatta.

4 TESTISUUNNITELMAN ESITTELY

Virrankulutusta mitataan testisuunnitelman mukaisesti koko tuotteen elinkaaren ajan ja testisuunnitelma luodaan jokaiselle tuotteelle sopivaksi. Tyypillisesti mittaukset suoritetaan aina uuden ohjelmistoversion yhteydessä, mutta joskus voidaan suorittaa myös joitain tiettyjä osia koko testimatriisista. Taulukossa 1 on kuvattuna pieni osa V800-rannelaitteen testisuunnitelmasta, missä virrankulutusta mitataan laitteen eri kellotiloissa. Kyseisellä rannelaitteella käyttäjän on mahdollista muokata kellotilanäkymää haluamaansa muotoon. Kellotilaan on mahdollista asettaa pelkkä analoginen tai digitaalinen kellonäkymä tai sitten seuraavien tietojen yhdistelmä: päivämäärä, aika, omistaja tai aktiivisuus.

TAULUKKO 1. Testisuunnitelman ensimmäinen osa virrankulutus testauksessa

State		#1 iPhone	#2 Android	#3 iPhone	#4 Android
Diary 2min	mA				
Time (date, 10:00:00) "owner") **No activity 30min**	mA				
Time ("viisarikello") **No activity 30min**	mA				
Time 10:00 **Activity% Activity □ 60min**	mA				
Time (date, 10:00:00 "owner") **Activity □ 60min**	mA				
Time ("viisarikello") **Activity □ 60min**	mA				

Testisuunnitelman mukaan ensimmäinen testi on lyhytkestoinen (2 min) ja se suoritetaan laitteen ollessa täysin staattisessa tilassa, joko kalenteri- tai asetusvalikossa. Tämän jälkeen rannelaitteen virrankulutusta jatketaan mittaamalla sen kaikissa mahdollisissa kellotiloissa pidentämällä mittausaikaa ensin 30 minuuttiin ja tämän jälkeen vielä 60 minuuttiin. Testisuunnitelmaan sisältyy kellotilojen mittausten lisäksi myös testit esimerkiksi puhelimelta tulevien ilmoitusten virrankulutuksen mittaamiseen. Rannelaitteessa on paljon muitakin Bluetooth Low Energya (BLE) hyödyn-

täviä ominaisuuksia ilmoitusten lisäksi ja sen toimintaa mitataan todella laajasti testisuunnitelmassa. Tähän liittyviä mittauksia suoritetaan esimerkiksi sykemittauksessa, synkronoinnissa sekä Gopro-kameran ohjauksessa. Virrankulutusta mitataan lisäksi eri lajiprofiileissa, missä osassa tiloista käynnistyy myös Global Positioning System (GPS).

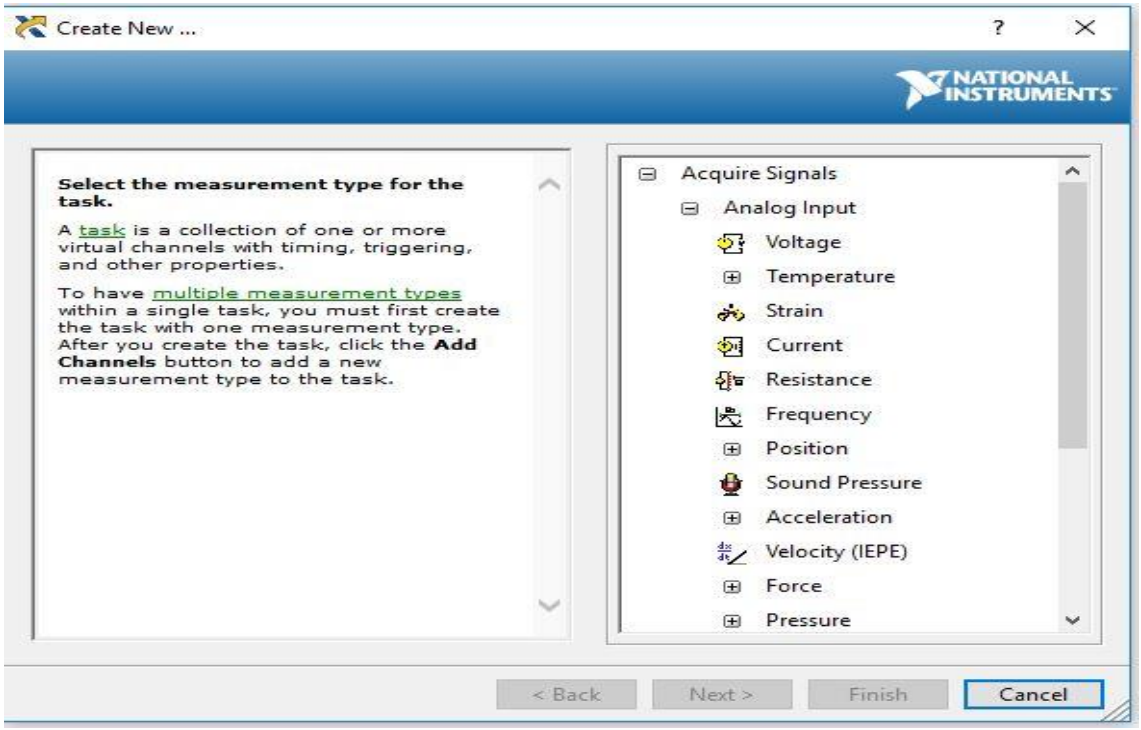
5 JÄRJESTELMÄN TOTEUTUS

Mallitoteutuksessa hyödynnetään useita eri ohjelmia, joista ainoastaan mittalaitteen edellyttämät ohjelmat on kuvattu tarkemmin tässä luvussa. Toteutuksessa on mittalaitteiston edellyttämien ohjelmien lisäksi asennettu IntelliJ IDEA -kehitysympäristö, jolla testitapausta ohjataan. Järjestelmään asennettiin myös Git- ja SourceTree -versionhallintaohjelmat.

Jotta mittalaitteella olisi mahdollista mitata virrankulutusta, on se aivan ensimmäiseksi saatava tunnistumaan käytettävässä tietokoneessa. Tietokoneen ja mittalaitteen välille saadaan muodostettua yhteys asentamalla mittalaitteen vaatima laiteajuri. Laiteajurin asennuksen yhteydessä tulevalta mittausohjelmistolta on mahdollista suorittaa mittausta paikallisesti mutta automaatiotarkoitukseen on myös luotava rajapinta laiteajurin sekä testiohjelman välille. Tässä työssä rajapinta on toteutettu Python x,y:llä.

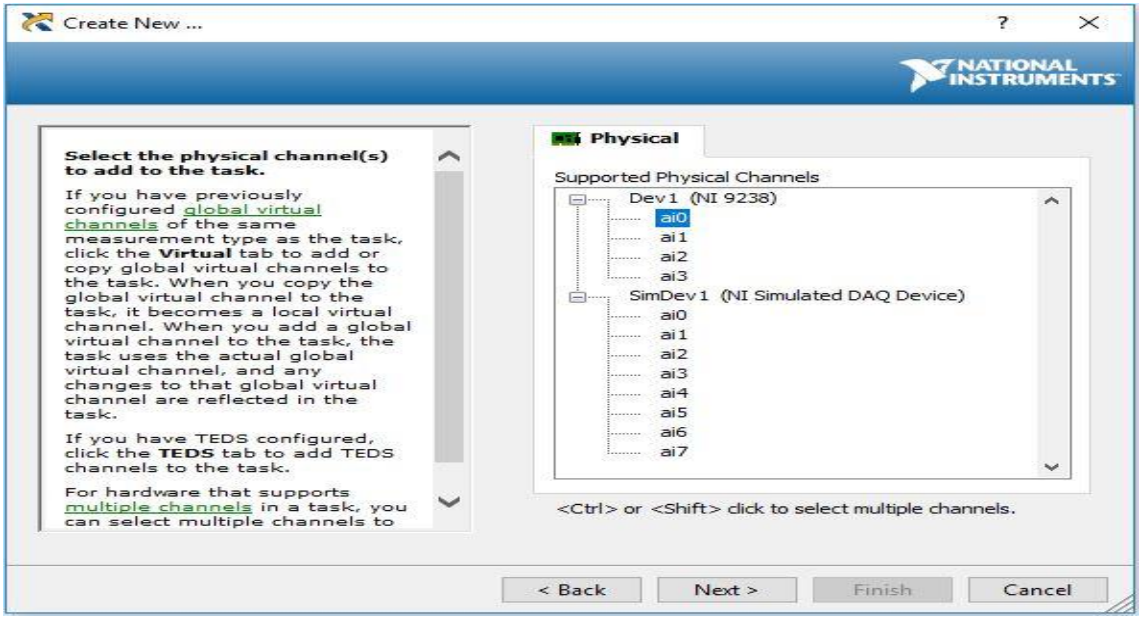
5.1 NI-DAQmx-ajurin asennus

National Instrumentsin mittalaitteille on saatavilla laitetoimittajan sivuilla laitteistoajurit, joiden avulla laitteen käyttöönotto ja myös päivitykset onnistuvat helposti. Paketti sisältää myös NI-MAX (Measurement & Automation Explorer) -mittausohjelman, joka mahdollistaa mittausten suorittamisen. Mitattavia määreitä analogista signaaleista (kuva 10) on valittavissa ohjelmasta todella laajasti. Mikäli käytössä on useampaa kanavaa mittaava laite, on mahdollista määritellä mittaukseen monikanavainen mittaus, jolloin on mahdollista suorittaa mittaus useammalle signaalille samanaikaisesti.



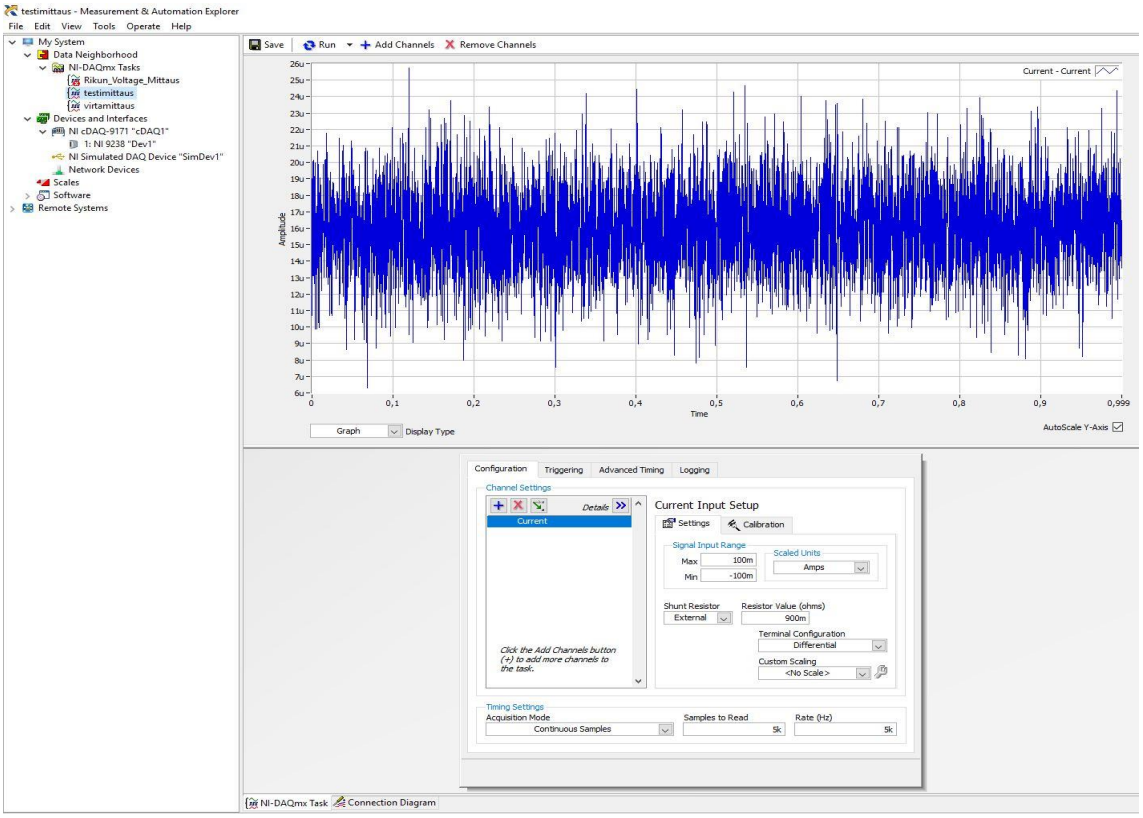
KUVA 10. NI MAX ja mittauksen luonti

Laiteajurin asennuksen yhteydessä testataan vielä, että laite, ajuri ja mittaohjelma toimivat oikein. Mittausohjelmasta on mahdollista käynnistää mittaus määrittelemällä ensin mitattava määre. Tässä tapauksessa valitaan virta. Tämän jälkeen tarvitsee ainoastaan määritellä mitattava kanava (kuva 11), johon mittaus kohdistuu, sekä mittauksen parametrit. Mittaohjelman näytöltä voidaan tarkastella mittaustuloksia joko graafisesti tai taulukkomuodossa (kuva 12).



KUVA 11. Mitattavan kanavan valinta

Mittausta voidaan seurata ohjelmassa olevalta graafiselta näytöltä, mutta sitä ominaisuutta ei ole mahdollista hyödyntää tässä työssä, koska mittaustuloksia käsitellään Python-skriptin avulla ainoastaan numeerisessa muodossa. Tulokset kuitenkin käännetään CSV-muotoon, minkä jälkeen testiautomaatioympäristö generoi tuloksista graafisen esityksen. (Kuva 12).



KUVA 12. Virtamittaus

5.2 Python(x,y):n sekä PyDAQmx:n ja nidaqmx:n asennus

Mallitoteutuksessa mittalaitteen ohjaus tapahtuu Python-kielellä ohjelmoidulla skriptillä, jossa käyttäjän on mahdollista määritellä ennen mittauksen suorittamista oikea mittalaite ja mittaukselle sopivat parametrit. Käyttäjä voi muuttaa esimerkiksi näytteistykseen sekä mittausaikaan liittyviä parametrejä ja myös nimetä mittauksesta saadut tulokset. Mallitoteutuksessa on hyödynnetty Python (x,y) -versiota (kuva 13).

Python (x,y)-versio sisältää sciPy- ja numPy-kirjastot, jotka mahdollistavat monisuuntaiset ja suuret taulukointiobjektit, matriisit sekä laajan kokoelman korkean tason matemaattisia funktioita.

Kirjastot onkin suunniteltu erityisesti tieteellisiin käyttötarkoituksiin ja mahdollistavat myös C-kielen käytön (7).

KUVA 13. Python (x,y)

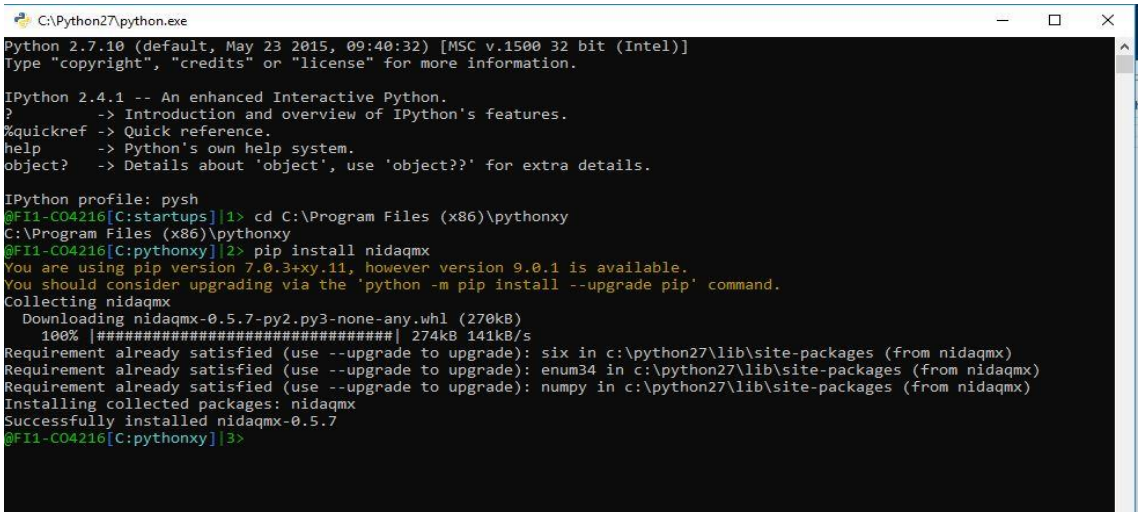
Jotta mittaukset voidaan automatisoida, pitää mittausohjelmistoa (NI-DAQmx) pystyä ohjaamaan Python-skripteillä. Rajapinta Pythonin ja mittausohjelmiston välille voidaan luoda ottamalla käyttöön Pythonin tulkissa PyDAQmx-moduuli, joka asennetaan komennolla: pip install PyDAQmx (kuva 14).

```
@FI1-CO4216[C:daqmx-master]|6> pip install PyDAQmx
You are using pip version 7.0.3+xy.11, however version 9.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
Collecting PyDAQmx
  Downloading PyDAQmx-1.4.1.tar.gz
Building wheels for collected packages: PyDAQmx
  Running setup.py bdist_wheel for PyDAQmx
  Stored in directory: C:\Users\TestNG\AppData\Local\pip\Cache\wheels\0b\77\10\83339cfa590dbbdcbc0ef56b236d8e34a8243b8bc24a363d55
Successfully built PyDAQmx
Installing collected packages: PyDAQmx
Successfully installed PyDAQmx-1.4.1
```

KUVA 14. PyDAQmx:n asennus

Moduulin lisäksi rajapinnan luomiseen tarvitaan Python API nidaqmx, joka asennetaan komennolla pip install nidaqmx (kuva 15). Python-pakettien asentaminen edellyttää, että pip (pakettihallintatyökalu) on asennettuna. Tämä voidaan tarkastaa komennolla pip --version ja asentaa tarvitta-

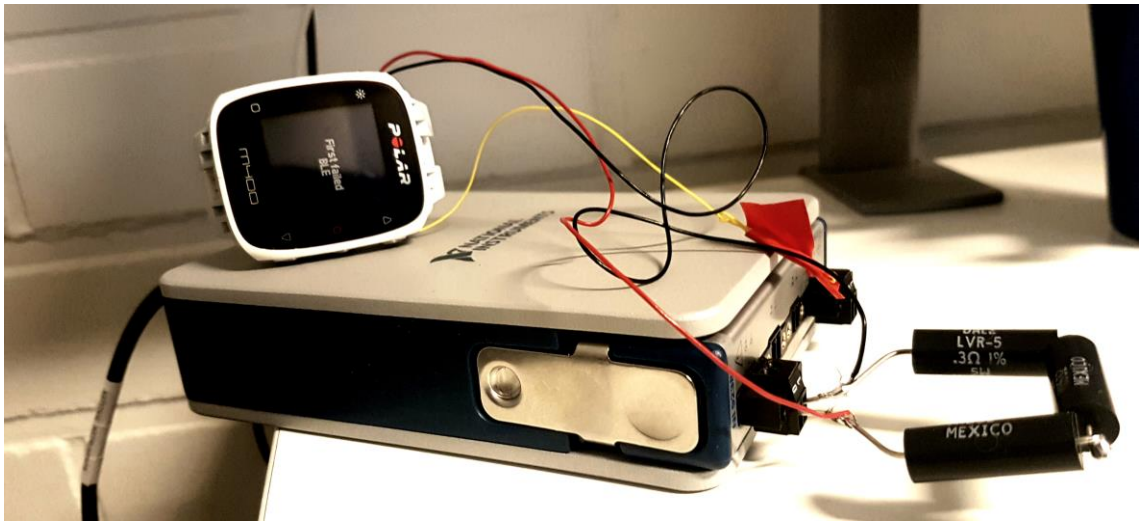
essa komennoilla get-pip.py ja python get-pip.py. Tässä työssä käytettävän tietokoneen käyttöjärjestelmä on Windows 10.



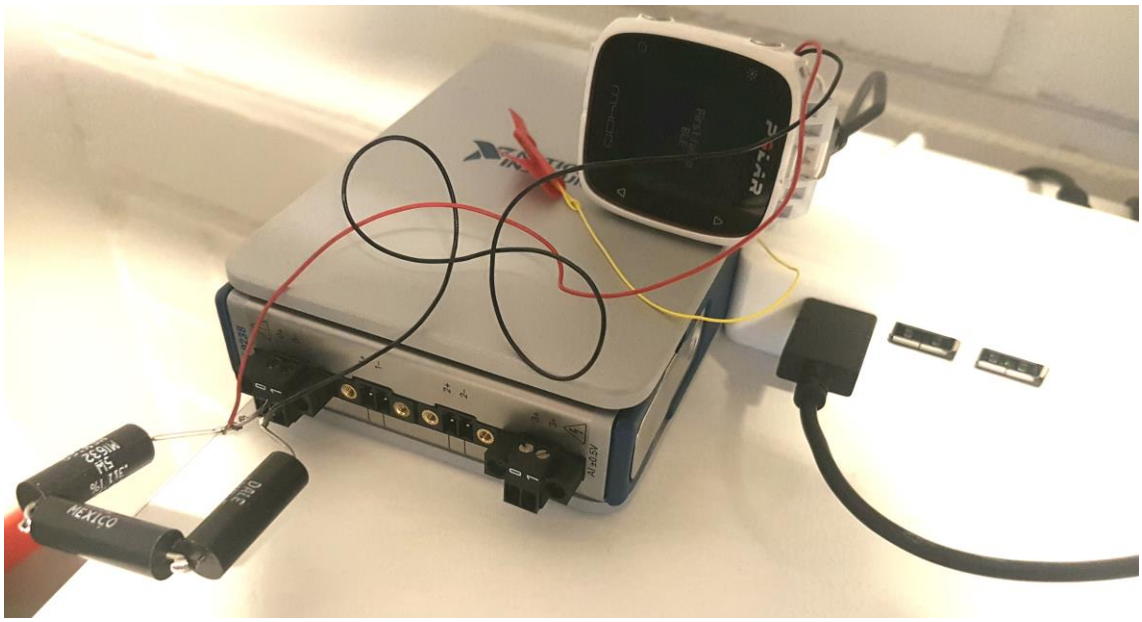
KUVA 15. nidaqmx API:n asennus

5.3 Mittausympäristön esittely

Mittalaite on kytketty tietokoneeseen USB-kaapelilla ja akulle johdotettu rannelaite (kuva 16) mittalaitteen kanavaan a0. Lisäksi rannelaite on kytkettynä USB-kaapelilla Yukush Yepkitiin (kuva 17). Mittalaitteeseen on myös kytketty kolme 0,3 ohmin vastusta, joiden yli virrankulutusta mitataan.



KUVA 16. Rannelaitteen kytkentä mittalaitteeseen



KUVA 17. Rannelaitteen kytkentä Yukush Yepkitiin

Onnistuneen laiteajurin, mittalaitteiston sekä Pythonin asennuksen jälkeen mittalaitetta on mahdollista ohjailla Python-skriptin avulla. Käytettävässä skriptissä laite mittaa näytteitä 1000 millisekunnin ajan kanavasta ai0. Mittaus tapahtuu 0.9 ohmin shunttivastuksen yli ja saaduista tuloksista tarkastellaan keskiarvoa (avg) laskevaa tulosta. Tulokset tallentuvat myös CSV-tiedostoksi. Esimerkkitapauksessa (kuva 18) M400-rannelaitteen virrankulutus akulta oli 640 µA.

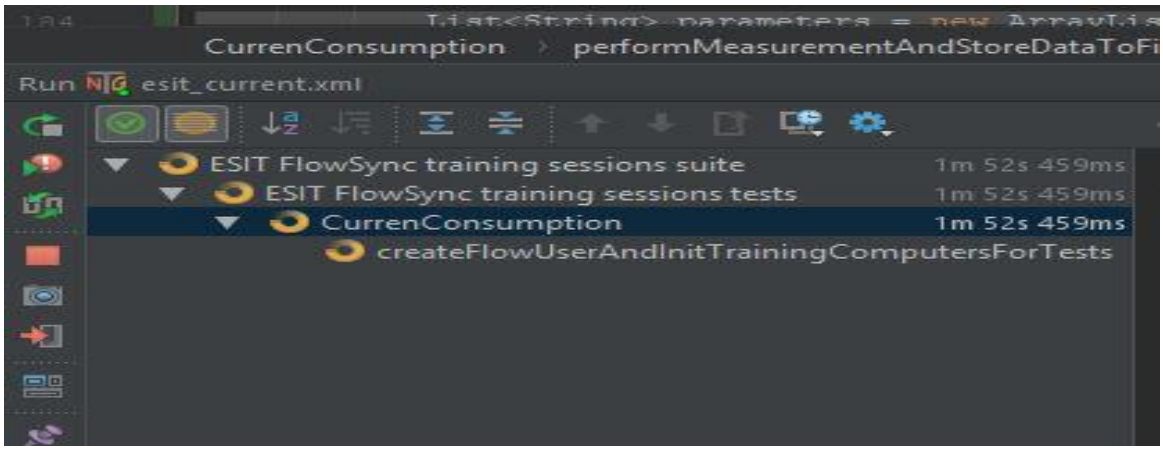
```
-----------------
@FI1-CO4216[C:debug]|132> python multiPowerMon6.py --current-chan "Dev1/ai0=Main" --sample-period 100 --write-period
Duration: 1000ms
Sample period: 100
Write period: 500ms (0.5sec)
CSV filename test.csv
Results filename result.txt
Samples/channel: 50
Resetting device: Dev1
Current Channel: Dev1/ai0, Limits: [-0.5 0.5], Shunt: 0.9
Current channels:
-----------------
Channel: Main Current (A)
Average:                        0.000640030955854
Median:                         0.000249393409796
Minimum:                        -0.000442533315022
Maximum:                        0.0120860050787
Std. Deviation:                 0.00191583241882
Variance:                       3.670413857e-06
---------------

------------------------------------------------------
Voltage channels:
```

KUVA 18. Virranmittaus Pythonilla

6 MITTAUS UUDELLA JÄRJESTELMÄLLÄ

Mallitoteutuksessa hyödynnetään National Instruments Compact DAQ -mittalaitetta, johon on yhdistetty YKUSH-YepKit, jonka avulla voidaan ohjata rannelaitetta USB-yhteyden avulla. YepKit mahdollistaa myös useamman laitteen kytkennän samanaikaisesti. Mittalaite on yhteydessä tietokoneeseen, johon on asennettu mittalaitteen ajurit sekä Python mittausten suorittamiseksi. Mittaus käynnistetään testiautomaatioympäristöön yhdistetyltä kehitystyökalulta (IntelliJ IDEA) käynnistämällä mittaukseen soveltuva testitapaus (kuva 19). Javalla ohjelmoidussa testitapauksessa ohjataan rannelaite haluttuun tilaan sekä kutsutaan mittalaitetta ohjaavaa Python-skriptiä, jolla puolestaan käynnistetään virrankulutusmittaus rannelaitteelta.

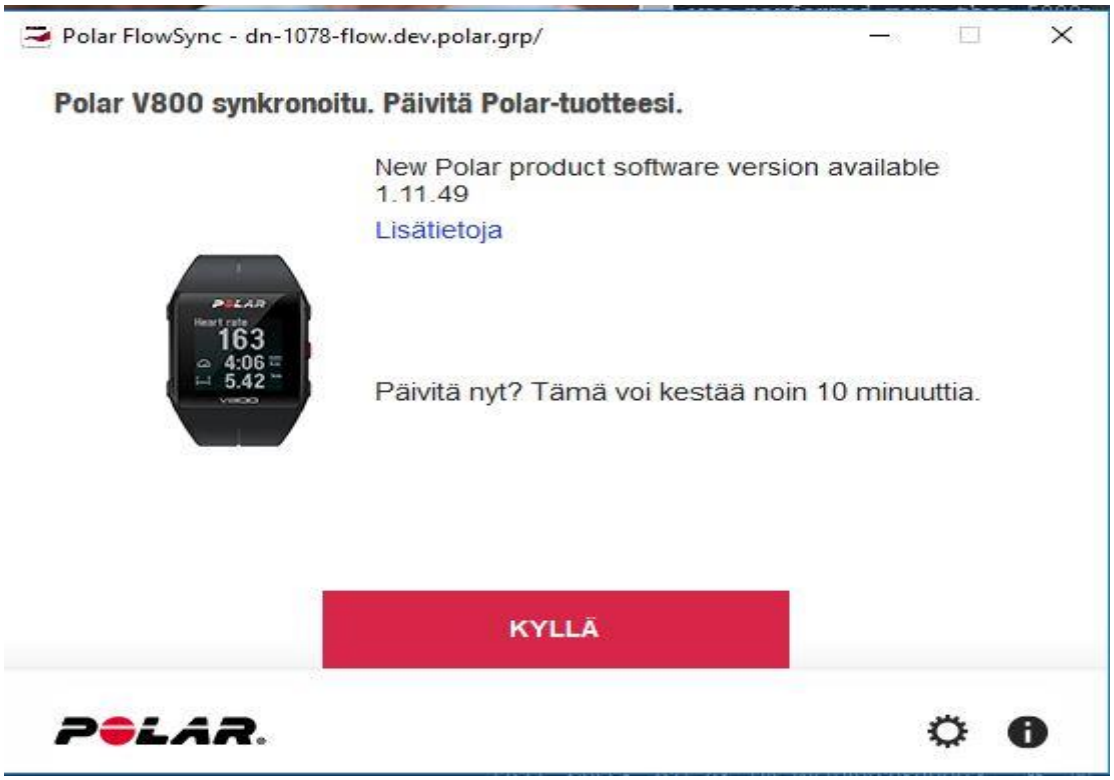
Testitapauksen käynnistyttyä rannelaite yhdistyy Polar Flow Sync -palveluun, missä suoritetaan automatisoidusti laitteen käyttöönotto. Käyttöönotossa palveluun luodaan käyttäjä, minkä jälkeen rannelaite synkronoidaan palveluun luodulla käyttäjädatalla sekä alustetaan käyttöä varten (kuva 20). Rannelaitteen näytöltä on myös mahdollista tarkastella, milloin synkronointi on suoritettu onnistuneesti loppuun (kuva 21). Käyttöönoton jälkeen testitapaus käynnistää harjoitusdatan synkronoinnin rannelaitteelta palveluun. Ennen mittausvaiheen käynnistymistä rannelaitteelta testitapaus tarkastaa vielä rannelaitteelta ohjelmistoversion ja päivittää sen automaattisesti, mikäli laitteella on vanhentunut ohjelmistoversio (kuva 22). Päivityksen jälkeen rannelaite käynnistyy kellotilaan, eikä rannelaitteen taustalla pitäisi olla tästä syystä muita prosesseja käynnissä. Rannelaiteelle on nyt mahdollista aloittaa itse virrankulutuksen mittaaminen.



KUVA 19. Testitapauksen käynnistäminen

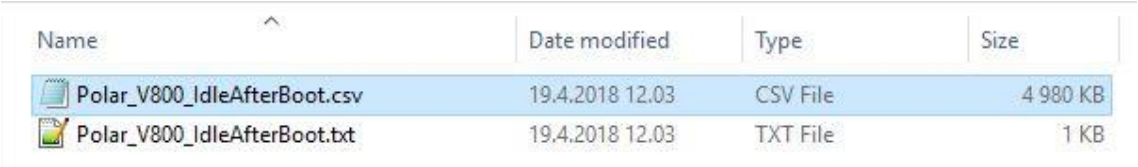
KUVA 20. Rannelaitteen synkronointi

KUVA 21. Synkronointi valmis



KUVA 22. Rannelaitteen päivitys

Testattava laite on nyt saatu haluttuun tilaan ja testitapaus käynnistää ensiksi noin minuutin kestävän stabilointivaiheen, minä aikana rannelaitteen USB-yhteys katkaistaan, ettei se ei aiheuttaisi häiriötä tai vääristäisi mittaustuloksia. Rannelaitteen stabiloiduttua, käynnistyy virrankulutusmittaus, joka tässä työssä toteutetaan rannelaitteen kellotilassa, missä lisäksi näkyy laitteen omistajan tiedot. Testitapauksessa kutsutaan mittaustyökalua ohjaavaa Python-skriptiä, joka puolestaan käynnistää virtamittauksen rannelaitteelta. Mittauksen kestoa voidaan vaihdella eri tapauksissa, mutta tässä työssä mittauksen pituus on 33 minuuttia. Mittausajan tultua täyteen, mittaus katkeaa ja tulokset tallentuvat CSV- sekä TXT-tiedostoina koneen muistiin (Kuva 23). TXT-tiedostoon tallentuvat jokainen näyte mittauksen ajalta. Näytteitä mittauksessa otetaan 100 millisekunnin pituisina, aina 500 millisekunnin välein. TXT-tiedostosta on luettavissa suoraan mittauksen tulokset, joista ilmenee mm. mittauksen keskiarvo, mediaani sekä varianssi (kuva 24).



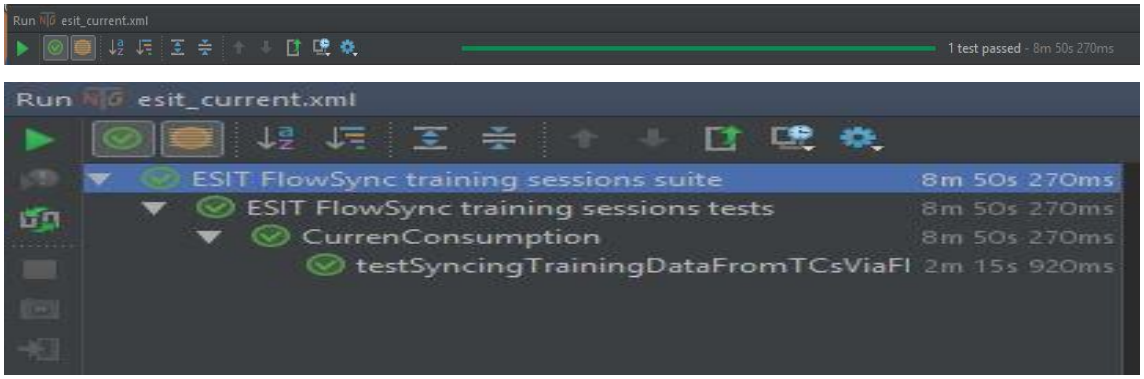
KUVA 23. Mittaustulokset

Polar_V800_IdleAfterBoot.txt | Polar_V800_IdleAfterBoot.csv | result.txt

```
1  Current channels:
2  -----------------
3  Channel: Main Current (A)
4  Average:            0.0173025028917
5  Median:             0.000239385863333
6  Minimum:            -0.00483435748602
7  Maximum:            0.105465676755
8  Std. Deviation:         0.0250444543307
9  Variance:           0.000627224692724
10 ----------------
11
12 ------------------------------------------------------
13 Voltage channels:
14 -----------------
15
```

KUVA 24. Mittaustulosten yhteenveto

Tulosten tallennettua testitapaus kytkee vielä USB-yhteyden takaisin rannelaitteelle ja lopettaa testin suorittamisen. Kehitysympäristöstä on tarkistettavissa, menivätkö testi ja testitapauksen kaikki vaiheet onnistuneesti läpi. Testivaiheita kuvaavat vihreät pallot ovat merkkinä siitä, että kaikki testitapauksen vaiheet menivät läpi onnistuneesti (Kuva 25). Mikäli jokin palloista on punainen, on se merkkinä testivaiheen epäonnistuneesta suorittamisesta ja testi on tällöin suoritettava uudelleen.



KUVA 25. Onnistunut testi

7 JATKOKEHITYS

Vakaasti toimivaan mallitoteutukseen on mahdollista lisätä toiminnallisuutta todella paljon ja parantaa testiautomaatiolla saatuja hyötyjä entisestään. Projektin aikana ilmeni muutamia selkeitä jatkokehityskohteita, joiden käyttöönotto olisi erityisen suositeltavaa uudessa testiautomaatioympäristössä.

Jenkins mahdollistaa testitapausten automaattisen käynnistämisen ja testien ohjaamisen sinne asettavien lisäparametrien avulla. Sen avulla testit voidaan käynnistää johonkin tiettyyn kellon aikaan, toistaa niitä haluttu määrä sekä tarkastella testien tuloksia. Kun käytössä on useita eri testitapauksia, voidaan samalle laitteelle määrittää useita eri testejä ajettavaksi vaikkapa keskellä yötä ja testit voidaan myös käynnistää maantieteellisestä sijainnista riippumatta. Jenkins mahdollistaa myös testitulosten automaattisen graafisen esitystavan raportointijärjestelmässä. Jenkinsiin tallentuvat mittaustulokset CSV-muodossa on mahdollista siirtää raportointijärjestelmään luomalla rajapinta Jenkinsin ja raportointijärjestelmän välille. Tähän tarvitaan pääkäyttäjän oikeuksia.

Systeemiarkkitehtuuriin kuuluva aktiivisuussimulaattori sekä sitä ohjaavaa kontrolleria hyödynnetään myöhemmässä vaiheessa virrankulutuksen testaukseen. Tarkoituksena on kuitenkin luoda myös testitapaus virtamittaukseen rannelaitteen ollessa harjoitustilassa. Testitapauksessa mitataan virrankulutusta rannelaitteen ollessa liikkeessä ja simuloiden vaikkapa kävelyä. Aktiivisuussimulaattori, johon rannelaite kiinnitetään, on yhteydessä tietokoneeseen kontrollerin välityksellä ja simulaattorin ohjaus tapahtuu suoraan tietokoneelta.

Mallitoteutuksessa voidaan suorittaa mittausta ainoastaan yhdellä rannelaitteella kerrallaan. Laitteistoon on kuitenkin mahdollista lisätä yhteensä kolme rannelaitetta, mutta testitapauksesta puuttuu vielä useamman laitteen ohjaukset. Useamman laitteen samanaikaisessa ohjauksessa on myös huomioitavaa se, että mittalaite kykenee mittamaan virrankulutusta ainoastaan yhdestä rannelaitteesta kerrallaan, ja tämä onkin huomioitava testitapauksen suunnittelussa useammalle laitteelle.

Mallitoteutuksessa hyödynnytetyssä virtamoduulissa on neljä erillistä kanavaa, joista jokainen kanava on mahdollista ohjata mittaamaan rannelaitteen osavirtaa kuten Bluetoothia. Tämä ominaisuus on erityisen tärkeä silloin, kun halutaan etsiä juurisyy rannelaitteen kovaan virrankulutuk-

seen ja halutaan tarkastella yksittäisen moduulin käyttäytymistä. Mitattavan kanavan kanavamäärittelyt tehdään Python-skriptissä ja mittaus on mahdollista toteuttaa kaikille neljälle kanavalle samanaikaisesti. Kyseisiä virtamoduuleja on mahdollista myös kytkeä samaan mittausjärjestelmään useampia, jolloin on mahdollista kasvattaa mitattavien kanavien määrää merkittävästi.

Mittausympäristön huomioiminen jatkoa ajatellen olisi myös erityisen tärkeää. Mittausympäristö pitäisi saada mahdollisimman stabiiliksi ja ympäristötekijöistä johtuva vaihtelut mahdollisimman pieniksi. Mittausten suorittaminen täysin suljetussa ympäristössä minimoisi pienetkin ympäristöstä johtuvat vaihtelut ja mittaustulokset olisivat entistä vertailukelpoisempia.

8 YHTEENVETO

Työn tavoitteena oli automatisoida virrankulutusmittaus rannelaitetestauksessa. Ennen varsinaisen työn aloittamista oli selvitettävä tarkoin, mikä oli työn tarkoitus ja mitä etuja sillä haetaan yritykselle. Työlle asetettujen vaatimusten jälkeen oli mietittävä, kuinka se olisi toteutettavissa mahdollisimman kustannustehokkaasti, vieläpä niin, että mallitoteutusta olisi myöhemmin mahdollista laajentaa ilman suurempia ylläpidollisia rajoituksia. Testiautomaatioympäristönä oli alkuun tarkoitus hyödyntää Robot Frameworkia, joka olisi soveltunut mittausten automatisointiin kyllä hyvin, mutta ympäristön kartoituksen yhteydessä ilmeni, että työssä voitaisiin hyödyntää myös yrityksen jo olemassa olevaa automaatioympäristöä. Hyödyntämällä olemassa olevaa ympäristöä voitiin saavuttaa huomattavasti dynaamisempi lopputulos, koska se myös mahdollistaa tehokkaan jatkokehityksen. Jatkokehitystä tukee huomattavasti se, kun käytössä on sama järjestelmä ja samalla tavalla toteutetut testitapaukset kuin muissakin testiautomaatiota hyödyntävissä testeissä.

Työlle asetettujen vaatimusten lisäksi oli erityisen tärkeää selvittää olemassa oleva toimintaympäristö ja noudattaa jo käytössä olevia toimintamalleja niin paljon kuin mahdollista. Testiautomaation hyödyntäminen ja käyttöönotto uudessa ympäristössä oli järkevää miettiä yhteensopivaksi mahdollisimman pitkälle olemassa olevan testiautomaatioympäristön kanssa. Yhdenmukaiset toimintamallit sekä testiautomaatioympäristön hyödyntäminen uudessa toimintaympäristössä tuo mukanaan jonkin verran rajoituksia toimintamallien muodossa mutta antaa puolestaan valtavasti synergiaetuja jatkokehityshankkeitakin ajatellen. Suhteellisen vähän aikaa käytössä ollut, mutta täysin toimiva testiautomaatioympäristö tarjosi mittavan määrän valmiita kirjastoja, testitapauksia sekä skriptejä, joita olisi mahdollista hyödyntää työssäni soveltuvin osin. Toimivaan testiautomaatioympäristöön liittäminen mahdollistaisi lisäksi mittausten etäohjauksen ilmaan aikaan tai paikkaan liittyviä rajoituksia.

Koska virranmittaus oli tarkoitus integroida olemassa olevaan testiautomaatioympäristöön ja saada sitä kautta kaikki mahdollinen hyöty, oli se tehtävä täysin olemassa olevien määritysten mukaisesti. Käytössä oleva testiautomaatioympäristö on monimutkainen ja laaja kokonaisuus, jonka hahmottaminen kokonaisuudessaan on todella vaikeaa, mikäli ei työskentele päivittäin sen parissa ja tiedä sen kaikkia rajapintoja eri ohjelmiin ja resursseihin.

Virrankulutustestausta olisi ollut mahdollista automatisoida ainoastaan paikallisesti, jolloin jatkokehitys mahdollisuudet olisivat kuitenkin jääneet huomattavasti pienemmiksi. Integroimalla yksittäinen virrankulutustestaus osaksi koko testiautomaatioympäristöä saadaan käyttöön huomattavasti laajempi toimintaympäristö sekä toimintamallit tulevaisuutta ajatellen. Tulevaisuudessa onkin täysin mahdollista suorittaa jatkuvaa sekä pitkäkestoista testaamista ilman testaajan vaatimaa työpanosta tai maantieteellistä ja ajasta johtuvaa rajoitusta.

Täysin uudenlaisen mittauslaitteiston sekä testiympäristön käyttöönotto oli suhteellisen monimutkainen prosessi. Mittalaitteen ja mittausohjelman käyttöönotto tapahtui nopeasti, ilman mitään ongelmia. Suurimmat ongelmat muodostuivat mittalaitteen ohjauksesta Pythonilla, testiautomaatioympäristön konfiguroinnista sekä testitapauksen ohjelmoinnista. Mallitoteutuksessa on mahdollista suorittaa virrankulutusmittaus automatisoidusti rannelaitteen ollessa kellotilassa ja mittaustulos on mahdollista lukea tietokoneelle tallentuvasta tiedostosta. Alkuperäisen suunnitelman mukaan mittausta olisi mahdollista suorittaa useammalla laitteella samanaikaisesti ja tulokset tallentuisivat automaattisesti raportointijärjestelmään. Nämä ominaisuudet jäivät toteuttamatta tässä työssä mutta tulevat toteutumaan jatkokehitysvaiheessa.

LÄHTEET

1. Polar Electro Oy. 2017. Taloussanomat. Saatavissa: https://www.is.fi/yritys/polar-electro-oy/kempele/0209911-2/. Hakupäivä 3.1.2018.

2. Sipola Timo 2016. Sykemittarin keksijä ei jäänyt lepäämään laakereilleen – Polar porskuttaa hyvällä sykkeellä. Yle. Saatavissa: https://yle.fi/uutiset/3-8873583. Hakupäivä 3.1.2018.

3. cDAQ-9132 CompactDAQ Controller 2018. National Instruments. Saatavissa: http://www.ni.com/fi-fi/support/model.cdaq-9132.html. Hakupäivä 5.1.2018.

4. NI-9238 C Series Voltage Input Module 2018. National Instruments. Saatavissa: http://www.ni.com/fi-fi/support/model.ni-9238.html. Hakupäivä 5.1.2018.

5. YUKUSH 3. Yepkit USB 3.1 Switchable Hub. Yepkit Saatavissa: https://www.yepkit.com/product/300110/YKUSH3. Hakupäivä 5.1.2018.

6. High Voltage Power Monitor 2018. Monsoon Solutions Inc. Saatavissa: https://www.msoon.com/online-store. Hakupäivä 5.1.2018.

7. Device Specifications NI cDAQ™ -9132. NI CompactDAQ Four-Slot Controller 2014. National Instruments. Saatavissa: http://www.ni.com/pdf/manuals/371801b.pdf. Hakupäivä 7.1.2018

8. Frequently asked Questions 2018. SciPY.org Saatavissa: https://www.scipy.org/scipylib/faq.html#what-is-numpy. Hakupäivä 7.1.2018.

9. Mahesh Reddy 2015. What is TestNG. Built.io. Saatavissa: https://www.built.io/blog/what-is-testng. Hakupäivä 8.1.2018